slip 01

Q1. A) Write a program in GO language to accept user choice and print answers using arithmetic operators. [20 Marks]

```
package main
import "fmt"
func main(){
var n1 int
var n2 int
var add,sub,mul,div int
fmt.Println("Enter the n1:")
fmt.Scanln(&n1)
fmt.Println("Enter the n2:")
fmt.Scanln(&n2)
add=n1+n2
fmt.Println("Addition is:",add)
sub=n1-n2
fmt.Println("Substration is:",sub)
mul=n1*n2
fmt.Println("Multiplication is:",mul)
div=n1/n2
fmt.Println("Division is:",div)
}
```

OR

B) Write a program in GO language to accept n student details like roll_no, stud_name, mark1,mark2, mark3. Calculate the total and average of marks using structure.

```
package main
import "fmt"
type student struct{
roll int
m1,m2,m3 int
sname String
}
func main(){
var s1 student
var total,avg,n int
fmt.Println("Enter the no of student:")
fmt.Scanln(&n)
for i=0;i<n;i++{
fmt.Println("Enter the roll no:")
fmt.Scanln(&roll)
fmt.Println("Enter the name of student:")
fmt.Scanln(&sname)
fmt.Println("Enter the marks:")
fmt.Scanln(&m1,&m2,&m3)
}
for i=0;i<n;i++{
fmt.Println("Roll no=",s1.roll)
fmt.Println("Name=",s1.sname)
total=s1.m1+s1.m2+s1.m3
avg=total/3
fmt.Println("Total marks=",total)
fmt.Println("avg marks=",avg)
}
}
```
****

slip 02

Q1. A) Write a program in GO language to print Fibonacci series of nterms. [20 Marks]

```
package main

import "fmt"

func main(){
	var n int
	t1:=0
	t2:=1
	nextTerm:=0

	fmt.Print("Enter the number of terms : ")
	fmt.Scan(&n)
	fmt.Print("Fibonacci Series :")
	for i:=1;i<=n;i++ {
		if(i==1){
			fmt.Print(" ",t1)
			continue
		}
		if(i==2){
			fmt.Print(" ",t2)
			continue
		}
		nextTerm = t1 + t2
		t1=t2
		t2=nextTerm
		fmt.Print(" ",nextTerm)
	}
}
```

OR

B) Write a program in GO language to print file information. [20 Marks]

```
package main
import (
	"fmt"
	"os"
)
func main() {
	// Get the filename from command-line arguments
	args := os.Args[1:]
	if len(args) < 1 {
		fmt.Println("Usage: fileinfo <filename>")
		return
	}
	filename := args[0]

	// Get file information
	fileInfo, err := os.Stat(filename)
	if err != nil {
		fmt.Println("Error:", err)
		return
	}

	// Print file information
	fmt.Println("Name:", fileInfo.Name())
	fmt.Println("Size:", fileInfo.Size(), "bytes")
	fmt.Println("Mode:", fileInfo.Mode())
	fmt.Println("Modified Time:", fileInfo.ModTime())
	fmt.Println("Is Directory:", fileInfo.IsDir())
}
```

****

<u>slip 03</u>

Q1. A) Write a program in the GO language using function to check whether accepts number is palindrome or not.[20 Marks]

```
package main
import "fmt"
func main() {
	var number,remainder,temp int
	var reverse int = 0

		fmt.Print("Enter any positive integer : ")
	fmt.Scan(&number)

	temp=number
	for{
		remainder = number%10
		reverse = reverse*10 + remainder
		number /= 10

		if(number==0){
			break
		}
	}

	if(temp==reverse){
		fmt.Printf("%d is a Palindrome",temp)
	}else{
		fmt.Printf("%d is not a Palindrome",temp)
	}
}
```

OR

B) Write a Program in GO language to accept n records of employee information (eno,ename,salary) and display record of employees having maximum salary.

```
  package main
import "fmt"
type Employee Struct{
eno int
ename String
salary float
}
func main(){
var e1[10]   Employee
var n,a,max int
fmt.Println("enter the no of employee:")
fmt.Scanln(&n)
for i:=0;i<n;i++{
fmt.Println("Enter the employee no=")
fmt.Scanln(&e1.eno)
fmt.Println("Enter the emp Name=")
fmt.Scanln(&e1.ename)
fmt.Println("Enter the salary=")
fmt.Scanln(&e1.salary)
}
max=e1[0].salary
for i:=0;i<n;i++{
if(e1[i].salary>max){
max=e1.salary
a=i
}
}
fmt.Println("employee with maximum salary:")
fmt.Println("Emp no=",e1[a].eno,"emp name=",e1[a].ename,"emp salary=",e1[a].salary)
}
```

****

<u>slip 04</u>

Q1. A) Write a program in GO language to print a recursive sum of digits of a given number.

```
package main
import "fmt"
var sum int=0
func SumOfDigits(num int) int {
	if num > 0 {
		sum += (num % 10) //add digit into sum
		SumOfDigits(num / 10)
	}
	return sum
}

func main() {
	var num int = 0
	var result int = 0

	fmt.Printf("Enter number: ")
	fmt.Scanf("%d", &num)

	result = SumOfDigits(num)

	fmt.Printf("Sum of digits is: %d¥n", result)
}
```
OR

B) Write a program in GO language to sort array elements in ascending order.

```
package main
import "fmt"
func main() {
	var arr [5]int
		var n int
	var min int = 0
	var temp int = 0

	fmt.Printf("Enter array elements: ¥n")
	for i := 0; i <= 4; i++ {
		fmt.Printf("Elements: arr[%d]: ", i)
		fmt.Scanf("%d", &arr[i])
	}

	for i := 0; i <= 4; i++ {
		min = i
		for j := i + 1; j <= 4; j++ {
			if arr[j] < arr[min] {
				min = j
			}
		}
		temp = arr[i]
		arr[i] = arr[min]
		arr[min] = temp
	}

	fmt.Printf("Array after sorting: ¥n")
	for i := 0; i <= 4; i++ {
		fmt.Printf("%d ", arr[i])
	}
}
```
****

slip 05

Q1. A) Write a program in GO language program to create Text file [20 Marks]

```
package main
import "fmt"
import "os"
```

```
func main() {
	file, err := os.Create("Sample.txt")
	if err != nil {
		fmt.Println("Unable to open file: %s", err)
	}

	len, err := file.WriteString("Hello World")

	if err != nil {
		fmt.Println("Unable to write data: %s", err)
	}
	file.Close()

	fmt.Printf("%d character written successfully into file", len)
}
```
OR

B) Write a program in GO language to accept n records of employee information (eno,ename,salary) and display records of employees having minimum salary.

(repeat)slip no 03

****

slip 06

Q1. A) Write a program in GO language to accept two matrices and display its multiplication

```
package main
import "fmt"
func main() {
	var sum int = 0
	var matrix1 [2][2]int
	var matrix2 [2][2]int
	var matrix3 [2][2]int

	fmt.Printf("Enter matrix1 elements: ¥n")
	for i := 0; i < 2; i++ {
		for j := 0; j < 2; j++ {
			fmt.Printf("Elements: matrix1[%d][%d]: ", i, j)
			fmt.Scanf("%d", &matrix1[i][j])
		}
	}

	fmt.Printf("Enter matrix2 elements: ¥n")
	for i := 0; i < 2; i++ {
		for j := 0; j < 2; j++ {
			fmt.Printf("Elements: matrix2[%d][%d]: ", i, j)
			fmt.Scanf("%d", &matrix2[i][j])
		}
	}

	//Multiplication of matrix1 and matrix2.
	for i := 0; i < 2; i++ {
```

```
                for j := 0; j < 2; j++ {
                        sum = 0
                        for k := 0; k < 2; k++ {
                                sum = sum + matrix1[i][k]*matrix2[k][j]
                        }
                        matrix3[i][j] = sum
                }
        }

        fmt.Printf("Matrix1: ¥n")
        for i := 0; i < 2; i++ {
                for j := 0; j < 2; j++ {
                        fmt.Printf("%d ", matrix1[i][j])
                }
                fmt.Printf("¥n")
        }

        fmt.Printf("Matrix2: ¥n")
        for i := 0; i < 2; i++ {
                for j := 0; j < 2; j++ {
                        fmt.Printf("%d ", matrix2[i][j])
                }
                fmt.Printf("¥n")
        }

        fmt.Printf("Multiplication of matrix1 and matrix2: ¥n")
        for i := 0; i < 2; i++ {
                for j := 0; j < 2; j++ {
                        fmt.Printf("%d ", matrix3[i][j])
                }
                fmt.Printf("¥n")
        }
}
```

OR

B) Write a program in GO language to copy all elements of one array into another using a method.

```
 package main
 import "fmt"
 func main() {
     originalArray := []int{1, 2, 3, 4, 5}
     copyArray := make([]int, len(originalArray))

     copy(copyArray, originalArray)

     fmt.Println("Original Array: ", originalArray)
     fmt.Println("Copy Array: ", copyArray)
}
```

****

slip 07

Q1. A) Write a program in GO language to accept one matrix and display

its transpose. [20 Marks]

```
package main
import "fmt"
func main() {
	var i, j, rows, columns int

	var orgMat [10][10]int
	var transposeMat [10][10]int

	fmt.Print("Enter the Matrix rows and Columns = ")
	fmt.Scan(&rows, &columns)

	fmt.Println("Enter Matrix Items to Transpose = ")
	for i = 0; i < rows; i++ {
		for j = 0; j < columns; j++ {
			fmt.Scan(&orgMat[i][j])
		}
	}
	for i = 0; i < rows; i++ {
		for j = 0; j < columns; j++ {
			transposeMat[j][i] = orgMat[i][j]
		}
	}
	fmt.Println("* The Transpose Matrix Items are *")
	for i = 0; i < columns; i++ {
		for j = 0; j < rows; j++ {
			fmt.Print(transposeMat[i][j], "    ")
		}
		fmt.Println()
	}
}
```

OR

B) Write a program in GO language to create structure student. Writea method show() whose receiver is a pointer of struct student.

```
package main
import (
	"fmt"
)
// Define a structure for student
type student struct {
	name     string
	rollNo int
	grade   string
}
// Define a method show() for student
func (s *student) show() {
	fmt.Printf("Name: %s¥nRoll No: %d¥nGrade: %s¥n", s.name, s.rollNo, s.grade)
}
func main() {
	// Create an instance of student
	s := student{"John Doe", 123, "A"}
	// Call the show() method on the student instance
	s.show()
}
```

****

slip 08

Q1. A) Write a program in GO language to accept the book details such as BookID, Title, Author, Price. Read and display the details of 'n' number of books. [20 Marks]

```
package main
import "fmt"
func main(){
var bid,n int
var title String
var Author String
var Price int
fmt.Println("Enter the no of book details=")
fmt.Scanln(&n)
fmt.Println("Enter the Book details=")
for i:=0; i<n; i++{
fmt.Println("Enter the book id=")
fmt.Scanln(&bid)
fmt.Println("Enter the book title=")
fmt.Scanln(&title)
fmt.Println("Enter the book Author=")
fmt.Scanln(&author)
fmt.Println("Enter the book price=")
fmt.Scanln(&Price)
}
fmt.Println("Detail of books..")
for i:=0; i<n; i++{
fmt.Println("book id=%d",bid)
fmt.Println("book title=%s",title)
fmt.Println("book Author=%s",author)
fmt.Println("Book Price=%d",Price)
}
}
```

OR

B) Write a program in GO language to create an interface shape that includes area and perimeter. Implements these methods in circle and rectangle type.

```
Package main
import "fmt"
type Shape interface {
		Area() float64
		Perimeter() float64
}
type Rectangle struct {
```

```
            Length, Width float64
}

func (r Rectangle) Area() float64 {
            return r.Length * r.Width
}

func (r Rectangle) Perimeter() float64 {
            return 2 * (r.Length + r.Width)
}
type Circle struct {
            Radius float64
}

func (c Circle) Area() float64 {
            return math.Pi * c.Radius * c.Radius
}

func (c Circle) Perimeter() float64 {
            return 2 * math.Pi * c.Radius
}

func main() {
        var s Shape = Rectangle {4.0,6.0}
        var s1 Shape = Circle {4.0}

        fmt.Printf("Area=%f,r.area
        fmt.Printf(" Perimeter = %f", r.Perimeter())

        fmt.Printf("Area = %f, c.Area())
  fmt.Println("Perimeter = %f", s1.Perimeter())
}
```

****

slip 09

Q1. A) Write a program in GO language using a function to check whether the accepted number is palindrome or not.

```
package main
import "fmt"
func main() {
            var number,remainder,temp int
            var reverse int = 0

                    fmt.Print("Enter any positive integer : ")
            fmt.Scan(&number)

            temp=number
            for{
                        remainder = number%10
                        reverse = reverse*10 + remainder
                        number /= 10
```

```
			if(number==0){
				break
			}
		}

		if(temp==reverse){
			fmt.Printf("%d is a Palindrome",temp)
		}else{
			fmt.Printf("%d is not a Palindrome",temp)
		}
}
```

OR
B) Write a program in GO language to create an interface shape that includes area and volume. Implements these methods in square and rectangle type. (Similar to slip 08)
****

slip 10

Q1. A) Write a program in GO language to create an interface and display its values with the help of type assertion.

```
package main
import "fmt"

func main() {

   // create an empty interface
   var a interface {}

   // store integer to an empty interface
   a = 10

   // type assertion
   interfaceValue := a.(int)

   fmt.Println(interfaceValue)
}
```

OR
B) Write a program in GO language to read and write Fibonacci series to the using channel.

```
  package main

import (
	"fmt"
)

func fibonacci(ch chan int, quit chan bool) {
	x, y := 0, 1
	for {
		select {
			case ch <- x: // write to channel ch
				x, y = y, x+y
			case <-quit:
				fmt.Println("quit")
				return
			}
		}
}

func main() {
		ch := make(chan int)
		quit := make(chan bool)
		n := 10

		go func(n int) {
			for i := 0; i < n; i++ {
				fmt.Println(<-ch) // read from channel ch
			}
			quit <- false
		}(n)

		fibonacci(ch, quit)
}
```

****

slip 11

Q1. A) Write a program in GO language to check whether the accept ed number is two digit or not.[20 Marks]

```
package main

import "fmt"

func main() {
	var num int
	fmt.Print("Enter a number: ")
	fmt.Scan(&num)

	if num >= 10 && num <= 99 {
		fmt.Println("The number is a two-digit number.")
	} else {
		fmt.Println("The number is not a two-digit number.")
	}
}
```

OR

B) Write a program in GO language to create a buffered channel, store few values in it and find channel capacity and length. Readvalues from channel and find modified length of a channel

```
  package main
import "fmt"

func main() {
	// Create a buffered channel with a capacity of 3
	ch := make(chan int, 3)

	// Store some values in the channel
	ch <- 1
	ch <- 2
	ch <- 3

	// Find the channel capacity and length
	capacity := cap(ch)
	length := len(ch)
	fmt.Printf("Channel capacity: %d, length: %d¥n", capacity, length)

	// Read the values from the channel
	fmt.Println(<-ch)
	fmt.Println(<-ch)
	fmt.Println(<-ch)

	// Find the modified length of the channel
	length = len(ch)
	fmt.Printf("Modified length: %d¥n", length)
}
```
****

slip 12

Q1. A) Write a program in GO language to swap two numbers using call by reference concept[20 Marks]

```
package main
import "fmt"
func swapByRef(x *int, y *int) {
	temp := *x
	*x = *y
	*y = temp
}

func main() {
	// Initializing two numbers
	num1 := 10
	num2 := 20

	// Printing original values
	fmt.Printf("Before swapping, num1 = %d and num2 = %d¥n", num1, num2)

	// Calling the swapByRef() function
	swapByRef(&num1, &num2)

	// Printing swapped values
	fmt.Printf("After swapping, num1 = %d and num2 = %d¥n", num1, num2)
}
```
OR

B) Write a program in GO language that creates a slice of integers, checks numbers from

the slice are even or odd and further sent to respective go routines through channel and display values
received by goroutines.

```
package main
import (
	"fmt"
	"sync"
)

func checkEvenOdd(number int, evenChan chan<- int, oddChan chan<- int, wg
*sync.WaitGroup) {
	defer wg.Done()

	if number%2 == 0 {
		evenChan <- number
	} else {
		oddChan <- number
	}
}

func main() {
	numbers := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
	evenChan := make(chan int)
	oddChan := make(chan int)
	var wg sync.WaitGroup

	for _, number := range numbers {
		wg.Add(1)
		go checkEvenOdd(number, evenChan, oddChan, &wg)
	}

	go func() {
		wg.Wait()
		close(evenChan)
		close(oddChan)
	}()

	fmt.Println("Even numbers:")
	for even := range evenChan {
		fmt.Println(even)
	}

	fmt.Println("Odd numbers:")
	for odd := range oddChan {
		fmt.Println(odd)
	}
}
```

****

slip 13

Q1. A) Write a program in GO language to print sum of all even and odd

numbers separately between 1 to 100.

```
package main
import "fmt"
func main() {
	evenSum := 0
	oddSum := 0

	for i := 1; i <= 100; i++ {
		if i % 2 == 0 {
			evenSum += i
		} else {
			oddSum += i
		}
	}

	fmt.Println("Sum of even numbers from 1 to 100 is", evenSum)
	fmt.Println("Sum of odd numbers from 1 to 100 is", oddSum)
}
```

OR

B) Write a function in GO language to find the square of a number and write a benchmark for it.

```
Package main
func square(n int) int {
	return n * n
}
import "testing"

func BenchmarkSquare(b *testing.B) {
	for i := 0; i < b.N; i++ {
		square(5)
	}
}
```

for run =>go test -bench=.

****

slip 14

Q1. A) Write a program in GO language to demonstrate working of slices (like append, remove, copy etc.) [20 Marks]

```
package main
import "fmt"
func main() {
	// Initialize a slice of integers
	numbers := []int{1, 2, 3, 4, 5}
	fmt.Println("Original slice: ", numbers)

	// Append a value to the slice
	numbers = append(numbers, 6)
	fmt.Println("After appending 6: ", numbers)

	// Remove an element from the slice
	index := 2
	numbers = remove(numbers, index)
	fmt.Println("After removing element at index ", index, ": ", numbers)

	// Copy one slice to another
	copiedNumbers := make([]int, len(numbers))
	copy(copiedNumbers, numbers)
	fmt.Println("Copied slice: ", copiedNumbers)
}
```

OR

B) Write a program in GO language using go routine and channel that will print the sum of the squares and cubes of the individual digits of a number. Example if number is 123 then
squares = (1 * 1) + (2 * 2) + (3 * 3)
cubes = (1 * 1 * 1) + (2 * 2 * 2) + (3 * 3 * 3).

```
package main
import (
	"fmt"
	"strconv"
)
func main() {
	num := 123
	squaresCh := make(chan int)
	cubesCh := make(chan int)

	go sumOfSquares(num, squaresCh)
	go sumOfCubes(num, cubesCh)

	squares := <-squaresCh
	cubes := <-cubesCh

	fmt.Printf("Sum of squares: %d¥n", squares)
	fmt.Printf("Sum of cubes: %d¥n", cubes)
}

func sumOfSquares(num int, ch chan int) {
	digits := getDigits(num)
	sum := 0

	for _, digit := range digits {
		square := digit * digit
		sum += square
	}

	ch <- sum
}

func sumOfCubes(num int, ch chan int) {
	digits := getDigits(num)
	sum := 0

	for _, digit := range digits {
```

```
            cube := digit * digit * digit
            sum += cube
        }

        ch <- sum
}

func getDigits(num int) []int {
        digits := []int{}
        str := strconv.Itoa(num)

        for _, char := range str {
            digit, _ := strconv.Atoi(string(char))
            digits = append(digits, digit)
        }

        return digits
}
```
****

slip 15

Q1. A) Write a program in GO language to demonstrate function return multiple values.[20 Marks]

```
package main
import "fmt"
func swap(x, y int) (int, int) {
      return y, x
}
func main() {
      // declaring two variables
      var a, b int = 10, 20

      // calling the `swap` function and assigning its return values to variables
      a, b = swap(a, b)

      // printing the swapped values of `a` and `b`
      fmt.Println("a =", a, " b =", b)
}
```

OR

B) Write a program in GO language to read XML file into structure and display structure

```
package main
import (
      "encoding/xml"
      "fmt"
      "os"
)
type Person struct {
      XMLName    xml.Name `xml:"person"`
      Name        string     `xml:"name"`
      Age          int          `xml:"age"`
```

```
	Address   Address   `xml:"address"`
}
type Address struct {
	Street     string `xml:"street"`
	City       string `xml:"city"`
	ZipCode   int       `xml:"zipcode"`
}
func main() {
	file, err := os.Open("person.xml")
	if err != nil {
		fmt.Println("Error opening XML file:", err)
		return
	}
	defer file.Close()

	var person Person
	decoder := xml.NewDecoder(file)
	err = decoder.Decode(&person)
	if err != nil {
		fmt.Println("Error decoding XML:", err)
		return
	}

	fmt.Printf("Name: %s¥n", person.Name)
	fmt.Printf("Age: %d¥n", person.Age)
	fmt.Printf("Street: %s¥n", person.Address.Street)
	fmt.Printf("City: %s¥n", person.Address.City)
	fmt.Printf("ZipCode: %d¥n", person.Address.ZipCode)
}
```
****

slip 16

Q1. A) Write a program in GO language to create a user defined package to find out the area of a rectangle.

// main.go file

```
package main

import (
	"fmt"
	"example.com/rectangle"
)

func main() {
	r := rectangle.Rectangle{Length: 10, Width: 5}
	area := rectangle.Area(r)
	fmt.Println("Rectangle:", r)
	fmt.Println("Area:", area)
}
```

// rectangle/rectangle.go file

```
package rectangle
```

```
type Rectangle struct {
	Length float64
	Width    float64
}

func Area(r Rectangle) float64 {
	return r.Length * r.Width
}
```
OR

B) Write a program in GO language that prints out the numbers from 0to 10, waiting between 0 and 250 ms after each one using the delay function.

```
package main
import (
	"fmt"
	"math/rand"
	"time"
)
func main() {
	for i := 0; i <= 10; i++ {
		fmt.Println(i)
		delay := time.Duration(rand.Intn(250)) * time.Millisecond
		time.Sleep(delay)
	}
}
```
****

slip 17

Q1. A) Write a program in GO language to illustrate the concept of returning multiple values from a function. (Add, Subtract, Multiply,Divide)[20 Marks]

```
package main
import "fmt"
func add(a, b int) (int, error) {
	return a + b, nil
}
func subtract(a, b int) (int, error) {
	return a - b, nil
}
func multiply(a, b int) (int, error) {
	return a * b, nil
}
func divide(a, b int) (float64, error) {
	if b == 0 {
		return 0, fmt.Errorf("cannot divide by zero")
	}
	return float64(a) / float64(b), nil
}
func main() {
	a := 10
	b := 5
	// call the add function and print the result
	result, err := add(a, b)
	if err != nil {
		fmt.Println("Error:", err)
	} else {
		fmt.Printf("%d + %d = %d¥n", a, b, result)
	}

	// call the subtract function and print the result
	result, err = subtract(a, b)
	if err != nil {
		fmt.Println("Error:", err)
	} else {
		fmt.Printf("%d - %d = %d¥n", a, b, result)
	}

	// call the multiply function and print the result
	result, err = multiply(a, b)
	if err != nil {
		fmt.Println("Error:", err)
	} else {
		fmt.Printf("%d * %d = %d¥n", a, b, result)
	}

	// call the divide function and print the result
	resultf, err := divide(a, b)
	if err != nil {
		fmt.Println("Error:", err)
	} else {
		fmt.Printf("%d / %d = %.2f¥n", a, b, resultf)
	}
}
```

OR

B) Write a program in GO language to add or append content at the end of a text file

```
package main
import (
	"bufio"
	"fmt"
	"os"
)
func main() {
	// Open the file for appending, creating it if it doesn't exist
	file, err := os.OpenFile("filename.txt", os.O_APPEND|os.O_CREATE|os.O_WRONLY,
0644)
	if err != nil {
		panic(err)
	}
	defer file.Close()

	// Prompt the user for the text to append to the file
	reader := bufio.NewReader(os.Stdin)
	fmt.Print("Enter text to append: ")
	text, err := reader.ReadString('¥n')
	if err != nil {
		panic(err)
	}

	// Write the text to the file
	_, err = file.WriteString(text)
	if err != nil {
		panic(err)
	}

	fmt.Printf("Successfully appended text to file '%s'¥n", file.Name())
}
```
****

slip 18

Q1. A) Write a program in GO language to print a multiplication table of number using function. [20 Marks]

```
package main
import "fmt"
func printMultiplicationTable(num int) {
	for i := 1; i <= 10; i++ {
		fmt.Printf("%d x %d = %d¥n", num, i, num*i)
	}
}
func main() {
	var num int
	fmt.Print("Enter a number to print multiplication table: ")
	fmt.Scan(&num)
	printMultiplicationTable(num)
}
```

OR

B) Write a program in GO language using a user defined package calculator that performs one calculator operation as per the user's choice.

```
package main
import (
	"fmt"
	"calculator"
)
func main() {
	var num1, num2 float64
	var choice string

	fmt.Println("Enter first number:")
	fmt.Scanln(&num1)

	fmt.Println("Enter second number:")
	fmt.Scanln(&num2)
	fmt.Println("Enter operation (+,-,*,/):")
	fmt.Scanln(&choice)

	switch choice {
		case "+":
			result := calculator.Add(num1, num2)
			fmt.Printf("Result: %.2f", result)
		case "-":
			result := calculator.Subtract(num1, num2)
			fmt.Printf("Result: %.2f", result)
		case "*":
			result := calculator.Multiply(num1, num2)
			fmt.Printf("Result: %.2f", result)
		case "/":
			result, err := calculator.Divide(num1, num2)
			if err != nil {
				fmt.Println("Error:", err)
			} else {
				fmt.Printf("Result: %.2f", result)
			}
		default:
			fmt.Println("Invalid choice")
	}
}
```
****

slip 19

Q1. A) Write a program in GO language to illustrate the function returning multiple values(add, subtract).

```
package main
import "fmt"
func addSubtract(a int, b int) (int, int) {
	return a+b, a-b
}
func main() {
	x, y := addSubtract(10, 5)
	fmt.Println("Sum:", x)
	fmt.Println("Difference:", y)
}
```
OR

B)

Write a program in the GO language program to open a file in READ only mode.

```
package main
import (
		"fmt"
		"os"
)
func main() {
		// Open the file in READ mode
```

```
	file, err := os.Open("example.txt")
	if err != nil {
		fmt.Println("Error:", err)
		return
	}
	defer file.Close()

	// Read the contents of the file
	data := make([]byte, 100)
	count, err := file.Read(data)
	if err != nil {
		fmt.Println("Error:", err)
		return
	}

	// Print the contents of the file
	fmt.Printf("Read %d bytes: %q¥n", count, data[:count])
}
```

****

slip 20

Q1. A) Write a program in Go language to add or append content at theend of a text file.[20 Marks]

```
package main
import (
	"fmt"
	"os"
)
func main() {
	// Open the file for appending. If the file doesn't exist, create it.
	file, err := os.OpenFile("example.txt", os.O_APPEND|os.O_CREATE|os.O_WRONLY,
0644)
	if err != nil {
		panic(err)
	}
	defer file.Close()

	// Write the new content to the end of the file.
	newContent := "This is new content that will be added to the end of the file."
	if _, err := fmt.Fprintln(file, newContent); err != nil {
		panic(err)
	}

	fmt.Println("Content added to the end of the file.")
}
```

OR

B) Write a program in Go language how to create a channel and illustrate how to close a channel using for range loop and close function. [20 Marks]

```
package main
import "fmt"
func main() {
    // Create a channel of integers with a buffer size of 3
    ch := make(chan int, 3)

    // Send some values to the channel
    ch <- 1
    ch <- 2
    ch <- 3

    // Close the channel
    close(ch)

    // Use a for range loop to read values from the channel
    for val := range ch {
        fmt.Println(val)
    }
}
```

****